

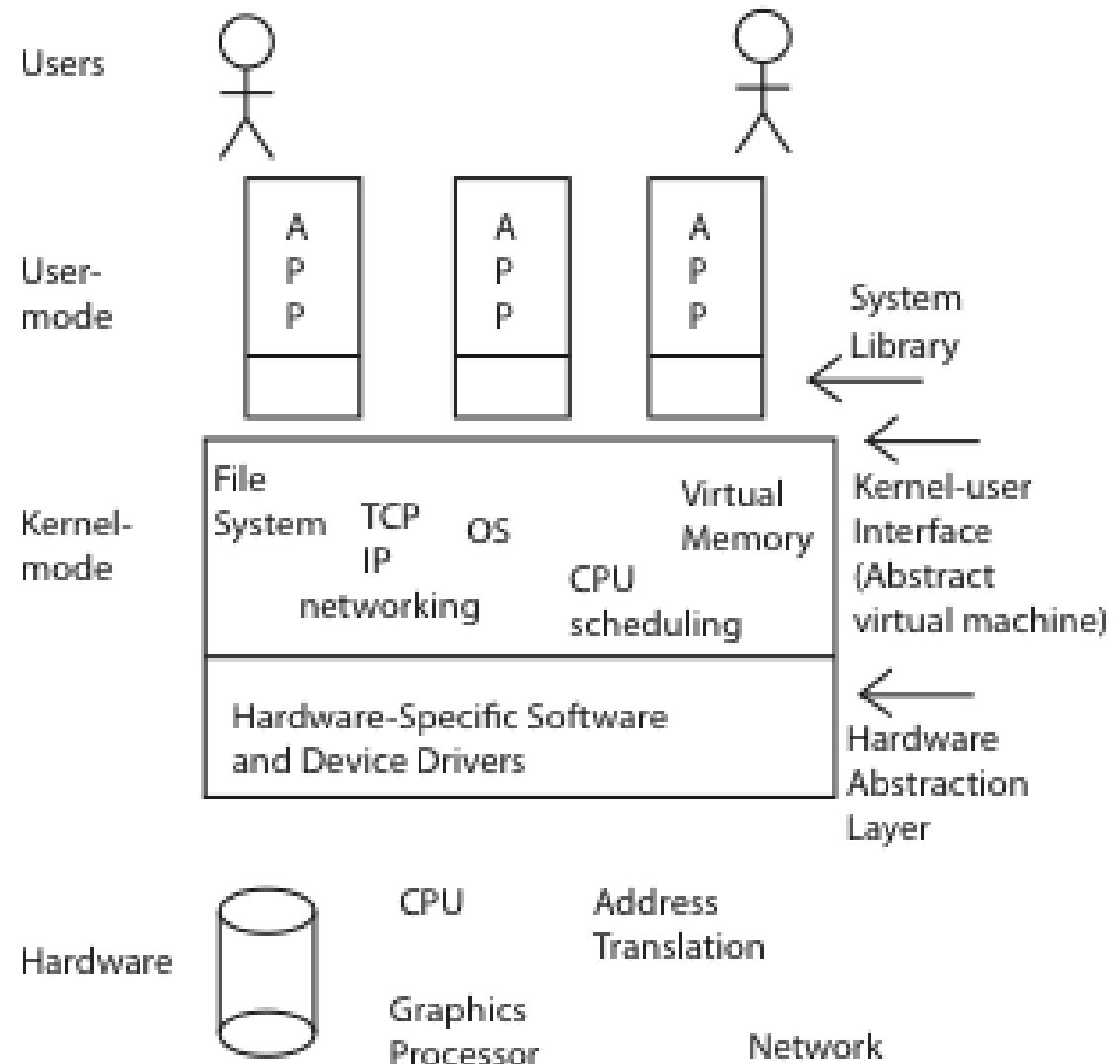
Operating Systems: Principles and Practice

Main Points

- Operating system definition
 - Software to manage a computer's resources for its users and applications
- OS challenges
 - Reliability, security, responsiveness, portability, ...
- OS history
 - How are OS X, Windows 7, and Linux related?

What is an operating system?

- Software to manage a computer's resources for its users and applications



Operating System Roles

- Referee:
 - Resource allocation among users, applications
 - Isolation of different users, applications from each other
 - Communication between users, applications
- Illusionist
 - Each application appears to have the entire machine to itself
 - Infinite number of processors, (near) infinite amount of memory, reliable storage, reliable network transport
- Glue
 - Provide common, standard services to applications
 - Simplifies application development
 - Libraries, user interface widgets, ...

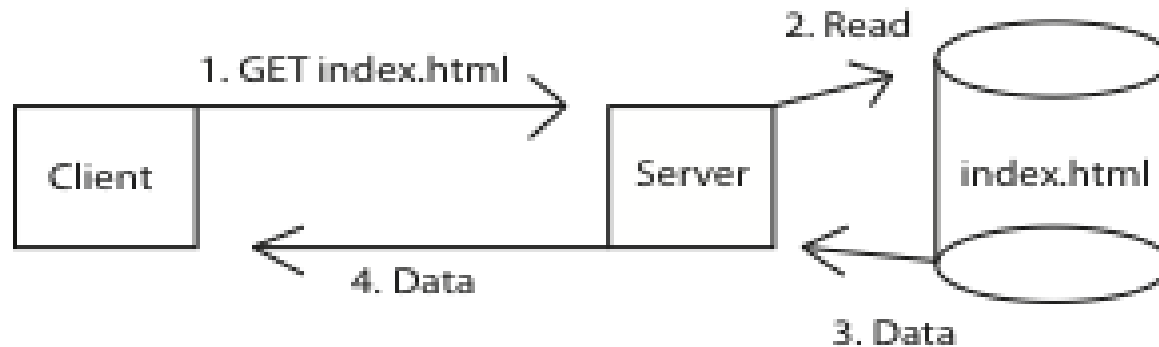
Operating system design patterns

- Cloud computing
 - Referee: how to allocate resources between competing applications in the cloud?
 - Illusionist: computing resources in a cloud evolve continuously, how to isolate applications from this evolution?
 - Glue: how to provide common, standardized access to the cloud services?
- Web services
 - Referee: ensure responsiveness when multiple tabs are opened at the same time
 - Illusionist: web services are geographically distributed for fault tolerance. Mask server failures to the users.
 - Glue: how does a browser achieve portable execution of scripts across different OS and HW platforms?

Operating system design patterns

- Multi-user database systems
 - Referee: how to enforce data access and privacy to different users ?
 - Illusionist: how to mask failures so that data remains consistent and available to users?
 - Glue: what common services to programs development?
- Internet
 - Referee: guarantee differentiated services to users and protect against DoS, spam, phishing etc...
 - Illusionist: internet appears as a unique, world-wide network but it is not!
 - Glue: internet protocols make applications independent of the underlying network architecture

Example: web service



- It defines an astonishingly simple behavior:
 - Receives a packet with a web page request
 - Retrieves the web page from disk
 - Sends back the page

Example: web service

However:

- Many requests involve data and computations
 - Think about search engines, a request may involve deep computations over large clusters of machines
- Multiple users issue requests at the same time
 - These must be managed simultaneously
- The server uses caches to speed up
 - Cache is shared among users, need for synchronized access mechanisms
- Servers send to clients scripts for pages customization
 - How does the client can protect itself from the execution of third party code that may embed viruses/spyware?

Example: web service

However:

- Web sites need to be updated
 - How to manage consistency with concurrent read requests?
- Client and server may run at different speeds
 - Need for speed decoupling
- Hardware supporting the web site may be updated
 - How to take advantage of this without rewriting the web server code?

OS Challenges

- Reliability
 - Does the system do what it was designed to do?
 - Availability
 - What portion of the time is the system working?
 - Mean Time To Failure (MTTF), Mean Time to Repair
- Security
 - Can the system be compromised by an attacker?
 - Privacy
 - Data is accessible only to authorized users
- Both require very careful design and code

OS Challenges

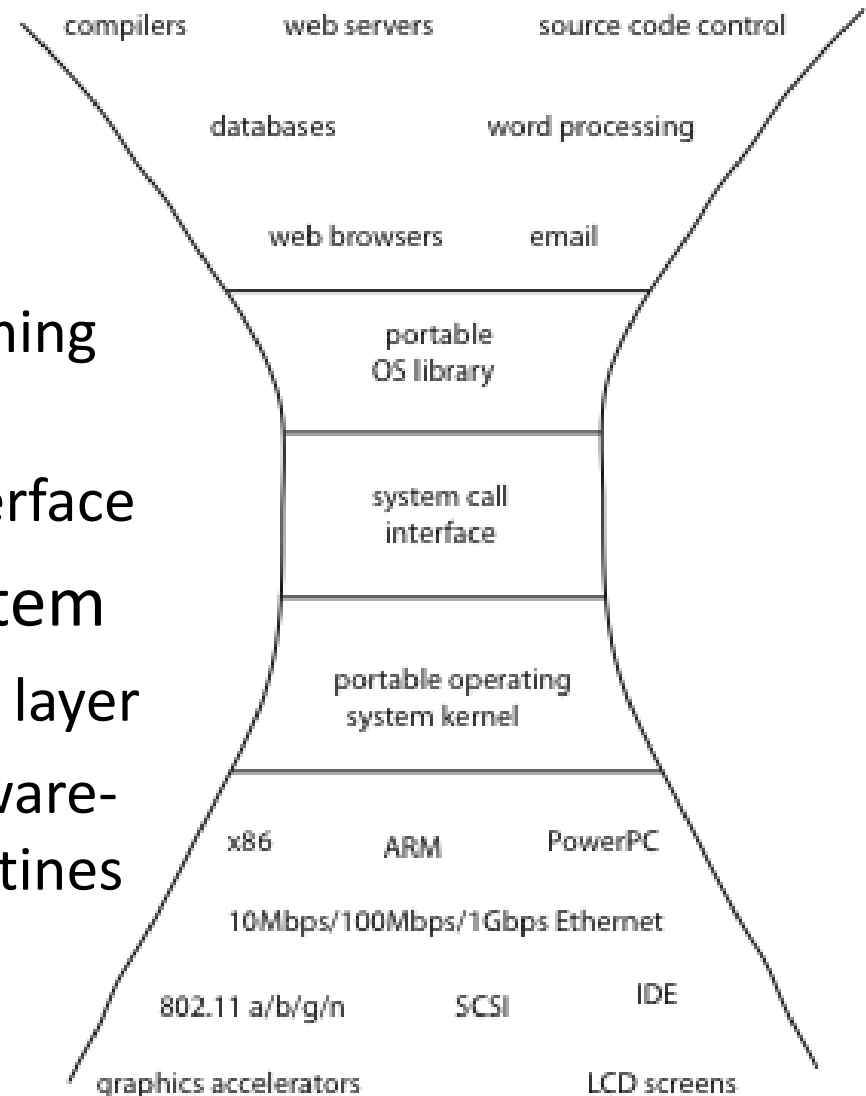
- Portability

- For programs:

- Application programming interface (API)
 - Abstract machine interface

- For the operating system

- Hardware abstraction layer
 - Pintos provides hardware-specific OS kernel routines



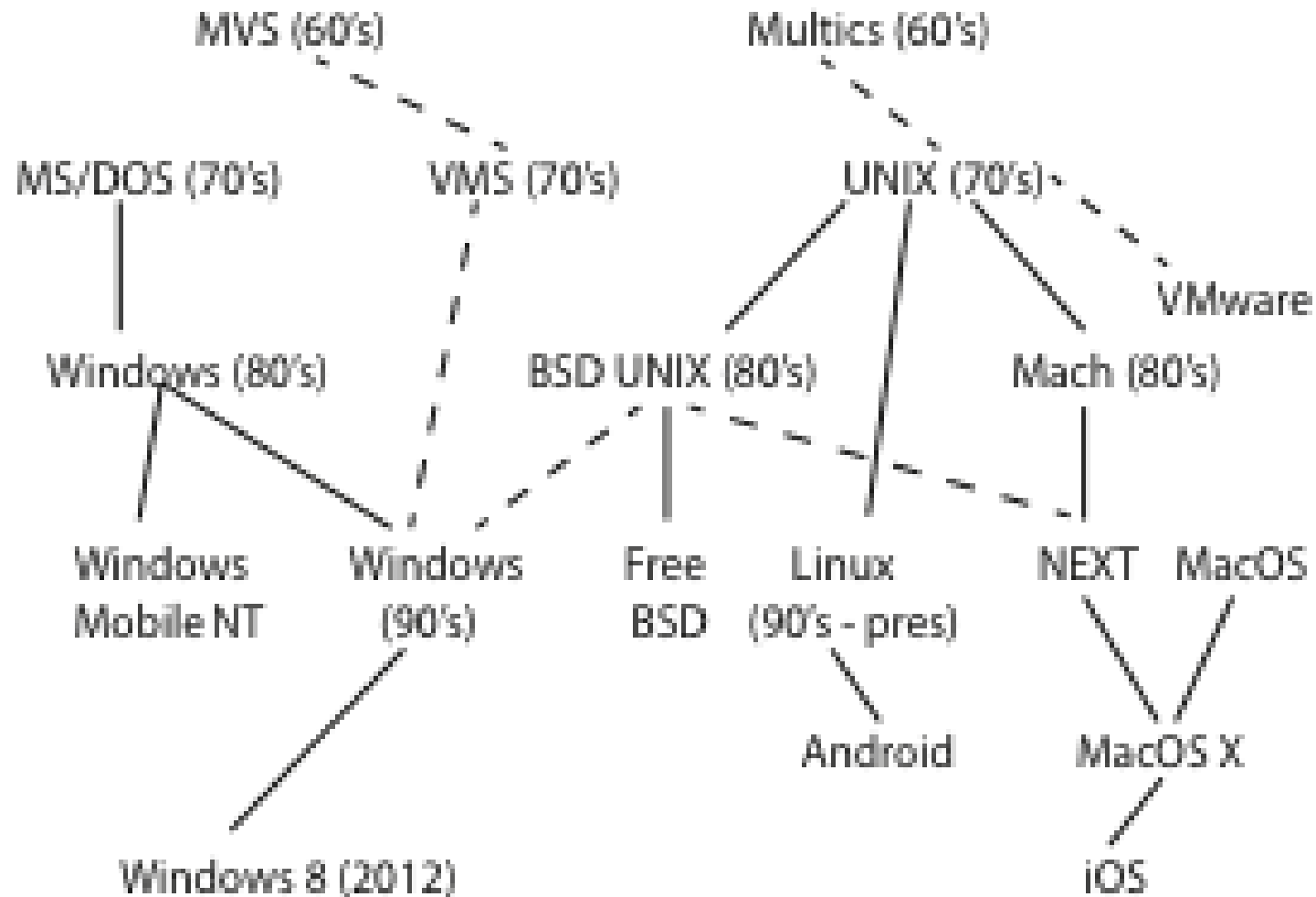
OS Challenges

- Performance
 - Latency/response time
 - How long does an operation take to complete?
 - Throughput
 - How many operations can be done per unit of time?
 - Overhead
 - How much extra work is done by the OS?
 - Fairness
 - How equal is the performance received by different users?
 - Predictability
 - How consistent is the performance over time?

OS Adoption

- Adoption is beyond control of an OS
 - Wide availability of applications
 - Wide availability of HW supporting it
- Network effect
 - App stores
 - Example: Android model vs iPhone model
- Proprietary vs open systems
 - Not a clear winner

OS History



- - - - - Influence

————— Descendant

Computer Performance Over Time

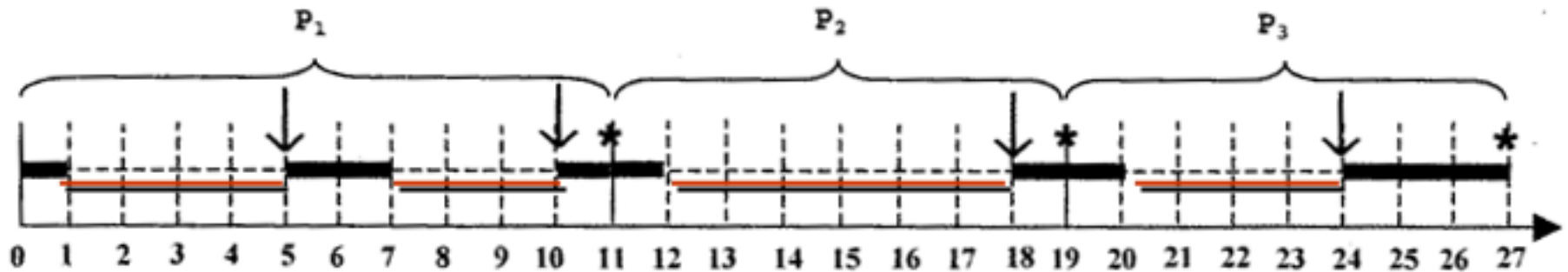
	1981	1996	2011	factor
MIPS	1	300	10000	10K
MIPS/\$	\$100K	\$30	\$0.50	200K
DRAM	128KB	128MB	10GB	100K
Disk	10MB	4GB	1TB	100K
Home Inter- net	9.6 Kbps	256 Kbps	5 Mbps	500
LAN network	3 Mbps (shared)	10 Mbps	1 Gbps	300
Users per machine	100	1	<< 1	100+

Early Operating Systems: Computers Very Expensive

- One application at a time
 - Had complete control of hardware
 - OS was runtime library
 - Users would stand in line to use the computer
- Batch systems
 - Keep CPU busy by having a queue of jobs
 - OS would load next job while current one runs
 - Users would submit jobs, and wait, and wait, and

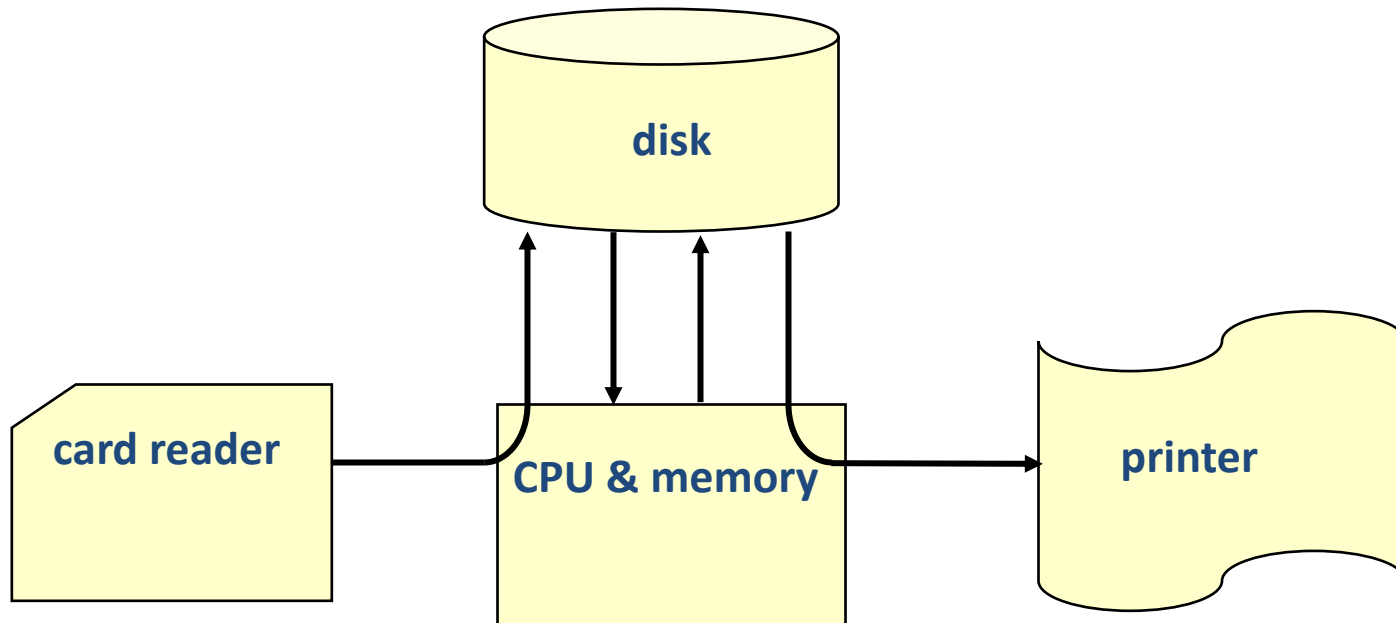
Single task systems

- Sequential execution



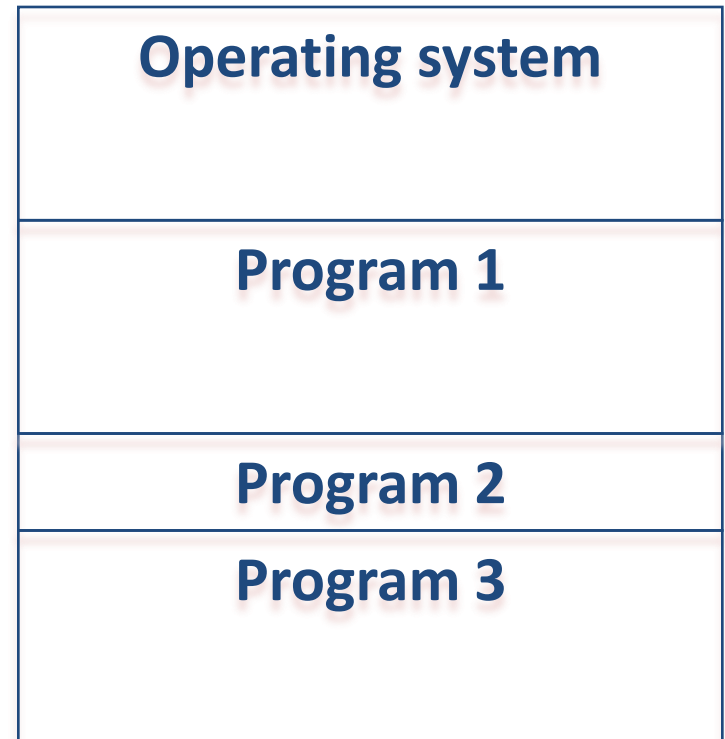
Early batch systems

- SPOOL: Simultaneous Peripheral Operation On-Line

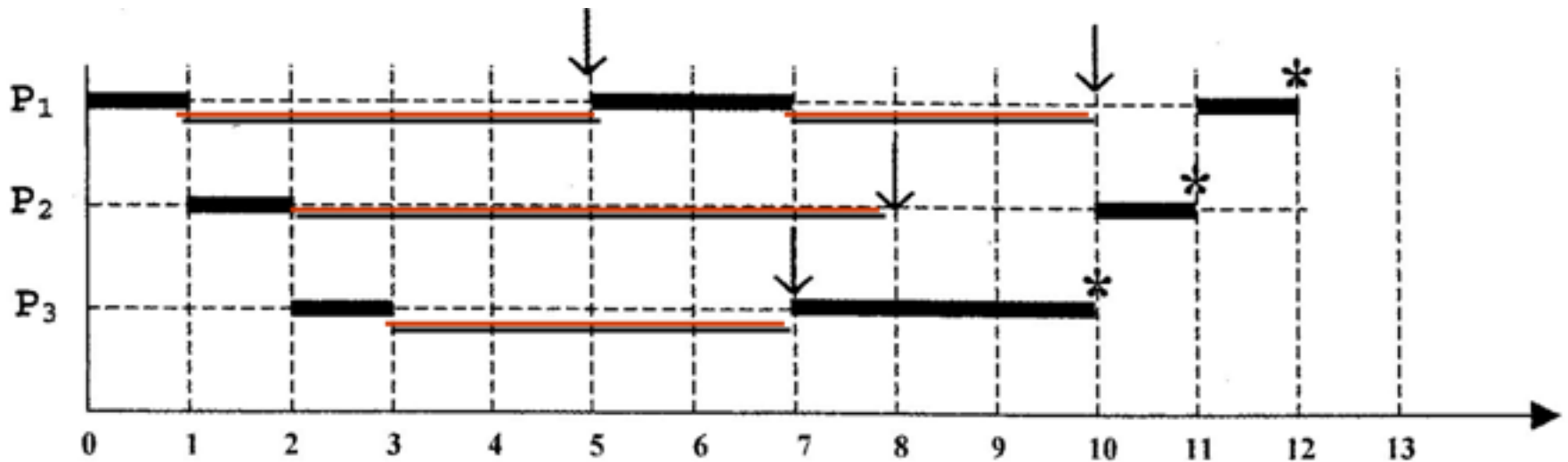
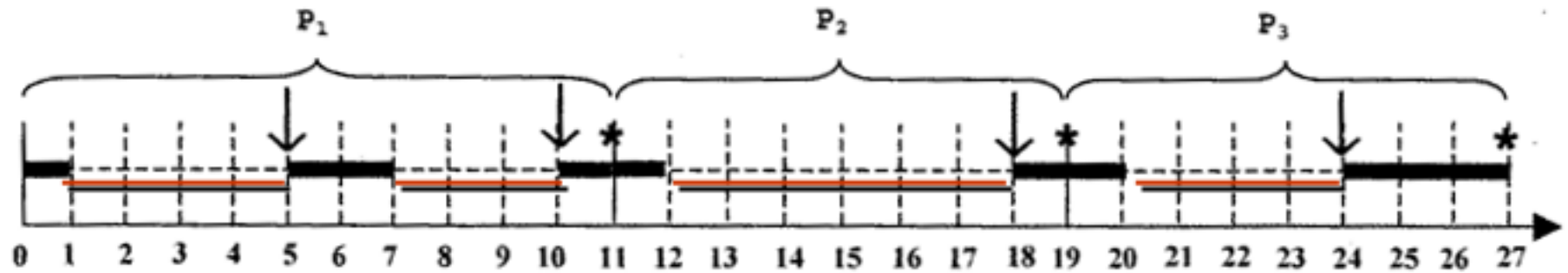


Multi-programmed batch systems

- multi-user system: several programs loaded in memory at the same time
- Spool optimization
- Resource optimization (processor, memory, devices)
 - Response time not important



Multi-tasking vs single-task

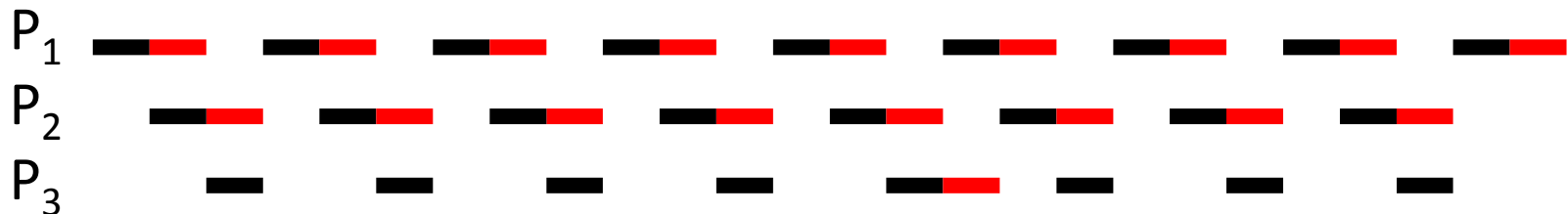
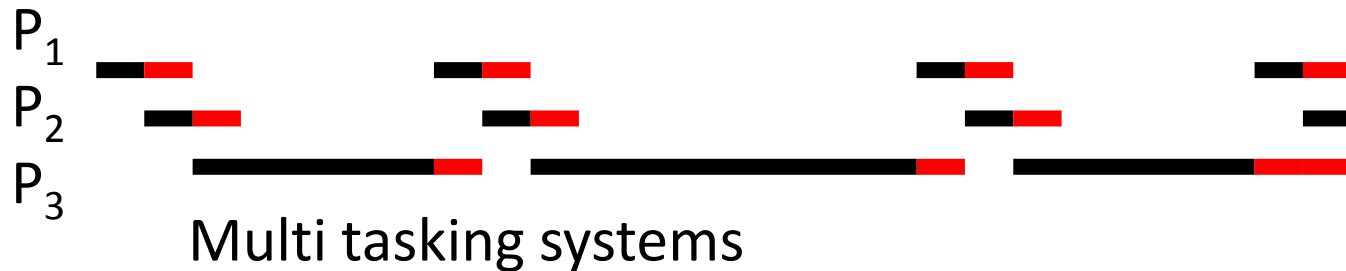


Time-Sharing Operating Systems: Computers and People Expensive

- Multiple users on computer at same time
 - Multiprogramming: run multiple programs at same time
 - Interactive performance: try to complete everyone's tasks quickly
 - As computers became cheaper, more important to optimize for user time, not computer time

Time-Sharing Operating Systems

- time sharing v.s. multitasking



Time sharing systems (QdT —)

Today's Operating Systems: Computers Cheap

- Smartphones
- Embedded systems
- Web servers
- Laptops
- Tablets
- Virtual machines
- ...

Tomorrow's Operating Systems

- Giant-scale data centers
- Increasing numbers of processors per computer
- Increasing numbers of computers per user
- Very large scale storage

Bonus Thought Question

- How should an operating system allocate processing time between competing uses?
 - Give the CPU to the first to arrive?
 - To the one that needs the least resources to complete? To the one that needs the most resources?
 - What if you need to allocate memory?
 - Disk?

Textbook

- Lazowska, Spring 2012: “The text is quite sophisticated. You won't get it all on the first pass. The right approach is to [read each chapter before class and] re-read each chapter once we've covered the corresponding material... more of it will make sense then. *Don't save this re-reading until right before the mid-term or final – keep up.*”